Document View

« Back to Results

< Previous Document 11 of 12 Next

Print | Email | Copy link | Cite this | Mark Document

Translate document from: Select language

Java Electronic Commerce Framework

Computer Reseller News. Manhasset: Sep 23, 1996. pg. 126

» Jump to indexing (document details)

Full Text (1770 words)

Editors note: The Java Electronic Commerce Framework is a framework for conducting business on the Internet and within intranets. The initial component is the JavaWallet, a client-side app that will be distributed as a core component.

his document introduces the Java Electronic Commerce Framework (JECF) and is organized as follows:

Background-What obstacles limit the growth of online commerce? How does Java help overcome these obstacles?

A Sample Transaction-What occurs during a typical purchase transaction in the JECF environment? What does the user see?

Architectural Overview-What is JECF? What are its compon-ents? What do they do? What security features are provided? How is access controlled?

Background

A typical Internet shopping application today must solve a number of potentially thorny problems. Due to a lack of standards, each shopping application must resolve these problems anew. Most existing applications rely on large, centralized computer servers which must have the capacity to accommodate large peak loads of simultaneous users. Using Java-based solutions, much of the computing currently performed on these monolithic back-end machines can be moved to the client platform, conserving expensive back-end resources and moving computing tasks closer to the end user. And, of course, since such a solution is Java-based, it will run on any Java-enabled platform.

As commercial use of the Internet grows, the need for a secure, standard mechanism for conducting commercial transactions becomes even greater.

In addition to existing payment instruments like credit and debit cards, this mechanism must also support emerging technologies such as electronic cash, electronic checks, and smart cash cards. JavaSoft's solution to this growing need is the Java Electronic Commerce Framework-a secure, extensible

framework for conducting business on the Internet.

A Sample Transaction

An online shopper using a Java-enabled Internet browser selects items for purchase from a page containing a shopping cart applet. When all of the desired items have been "placed in the shopping cart," the shopper clicks a button on the page that starts payment processing using the JECF. While most sellers will probably choose to implement their own shopping carts, a basic **shopping cart** implementation is included in the basic **Java** distribution.

When the Pay button on this page is pressed, the user's identity is verified and the user's private database of transaction information is opened.

The implementation that performs the functions of the payment protocol on the buyer's machine resides in a software module called a "cassette."

Cassettes are loaded locally if available, but they may also be loaded on demand from the Internet like applets. A payment cassette can send messages across the Internet according to its specific protocols. A cassette may read and store information in the database, and this information is protected from other cassettes as well as outsiders, i.e. the record of the user's actions is private.

The shopper next sees a payment page containing three distinct applets: an identity applet that establishes the identity of the seller; a tally applet that displays information about the goods and services to be purchased, including the total purchase price; and a payment instrument selection applet, which assists the buyer in the selection of a payment instrument that is accepted by the seller.

When the buyer has reviewed the details of the purchase, selected a payment instrument, and clicked the button that dismisses the payment window, a final confirmation page window appears. Unlike the payment page, which was constructed by the seller, this confirmation page is displayed by the JECF. This ensures, for example, that the amount of the purchase as seen by the JECF matches the amount previously displayed by the tally applet on the seller's payment page. Once the user confirms the transaction, the payment cassette initiates the actual payment process by transmitting its data to the appropriate remote server.

Depending on the protocol used by the payment cassette, the transaction may take several seconds or minutes to complete. During this period of time, information about the purchase is posted to the pending transactions list. If anything goes wrong during the payment processing (system crash, loss of network connectivity, etc.), this information can be used to "back out" of the transaction if necessary.

When all of the activities comprising the payment process have been completed, a verification page is displayed to the buyer to indicate that the transaction was completed successfully. Information about the transaction is removed from the pending transactions list, and saved in a permanent transaction register that allows the user to review past purchases in full detail.

Data in this register may also be made available (at the user's discretion, of course) as raw data to applications residing in service cassettes which provide services other than payment processing. One can imagine, for example, that the developers of a personal accounting package might choose to implement their software as a service cassette, and provide the capability to process data

in the transaction log using the same or a similar interface to an already familiar checkbook balancing program.

Such a service cassette could be sold (or, more properly, licensed) as a package, using JECF merely as a convenient distribution mechanism, or it might be paired with a payment cassette that charges the user on a per-use basis for the facilities of the service cassette.

In addition to the interfaces described above, the JECF also includes a series of configuration dialogs that are used to control and customize the JECF environment. These dialogs handle such activities as recording basic personal information, adding payment instruments to the environment, registering payment instruments with cassettes, etc.

Architectural Overview

The Java Electronic Commerce Framework (JECF) is a virtual point-of-sale device implemented in software that will be available for use by browsers and other Java-enabled environments. The JECF provides a framework for payment methods such as:

- Credit cards using the Secure Electronic Transactions (SET) protocol

- Smart cards Micro-transactions (pre-authorized payments for small amounts)

- Electronic checks

- Tokens for online games and services

- Frequent flier mileage and other types of incentive points

- Procurement cards

- Coupons

In addition to processing payments, JECF may be extended to provide other types of financial services, such as:

Financial analysis; Accounting; Budgeting; Tax reporting; Banking; Brokerage.

Simply stated, the goals of the JECF are:

Increase the volume of commercial transactions on the Internet.

Make purchasing goods and services a seamless part of the web-browsing process.

Establish a working marketplace that will provide an incentive to web-based authors and technologists to improve their content and technology.

Create an open, extensible platform in the most important areas of purchasing, banking, and personal and professional finance management.

From a structural perspective, the JECF consists of the following components:

Infrastructure: The architectural framework that defines the interactions of the

other JECF components listed below.

Database: A permanent record of a user's personal information, payment instruments, and transaction history.

Encryption support allows the database to be shared by different service providers who may either protect their data from one another or share it with one another as appropriate.

Payment cassettes: Modules that implement various protocols supported by individual payment instruments and service providers. When the user adds a new payment instrument to the environment, adding support for it is as easy as installing the appropriate payment cassette.

Service cassettes: Value-added services that build upon the basic JECF services, especially database services. Budgeting, financial analysis, and tax preparation are all excellent candidates for service cassette offerings.

Administrative Interfaces: Basic information about the user, the user's payment instruments, locally available payment and service cassettes, etc. is maintained using a series of configuration dialogs. These dialogs may also be used to customize certain aspects of the JECF's behavior and appearance.

Any application involved in moving funds around the Internet requires a secure environment. The JECF security model was developed by considering a wide range of potential attacks and devising specific strategies to defend against each of them. Security in the JECF is implemented at three levels, the first of which depends on the security facilities of Java 1.0 (which are discussed in greater detail in the Java White Paper):

Language features: Many basic Java language features are designed to disallow unsafe operations.

Runtime rules: Java's runtime rules also prevent unsafe uses of system resources.

Name space control: Strict partitioning of name spaces isolate functionality into discrete objects which may only share resources by design.

The following security features of Java 1.1 are also utilized by the JECF:

Signed applets: The use of signed applets allows digitally signed classes be trusted, and to perform certain operations which would otherwise be forbidden. The JECF also uses signed code to detect tampering with its code and to gain permission to some system resources.

Code verification: Before any Java code is executed, its compiled byte stream is inspected to ensure that it has not been tampered with and that it conforms to the runtime rules of the language specification.

The JECF extends these standard security facilities of the Java kernel by providing the following mechanisms:

User identification: A user name is unique in the JECF environment. Separate databases are established for multiple users who share the same machine. User names need not be globally registered.

User authentication: A password is used to authenticate a user before

permitting access to any JECF functionality or to the database. All password validation is performed locally, so passwords are never sent over any networks in any form.

User authorization: Payment cassettes are available to all users for appropriate use. The functionality provided by other cassettes may be made available only to specific users after authentication.

Network loader control: The basic Java mechanism for loading classes over the Internet is used.

Version numbers are included in the public name of classes to ensure that the correct version of code is loaded.

Local loader control: While executing JECF code, the Java CLASSPATH variable is set to a small list of root locations that will be searched for packages. Only the names of entry points which make correct checks are visible from outside a package.

Code and environment verification: A calling program asserts a role based on either its digital signature or on the role of a caller. Individual programs and cassettes check for proper state and sequencing.

Database access control: Access control lists specify which database objects should be made available to which cassettes, and a digital signature is required to read and write database objects.

Class invocation control: Which classes may invoke other classes is determined using digitally signed roles. A called class checks caller's role. Roles are mapped to rights on the called resources. Roles may also be delegated, including narrowing of access privileges.

Indexing (document details)

Subjects:	COMPUTERS, SOFTWARE
Section:	*White Paper – Sun's JECF*
Publication title:	Computer Reseller News. Manhasset: Sep 23, 1996. pg. 126
Source type:	Periodical
ISSN:	08938377
ProQuest document ID:	10632830
Text Word Count	1770
Document URL:	http://proquest.umi.com/pqdweb?did=10632830&sid=2&Fmt=7&clientId=19649&RQT=309&VName=PQD